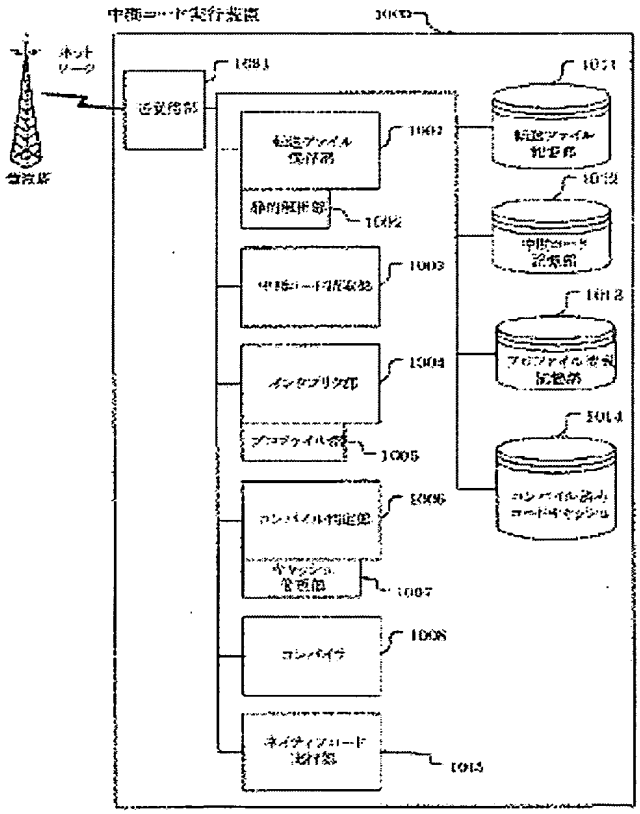
COMPUTER DEVICE AND PROGRAM EXECUTION METHOD

Patent number: JP2004118367
Publication date: 2004-04-15
Inventor: TAKAHASHI KATSUHIDE; KIKKO DAIZO; KIYOHARA RYOZO; NAKA KUNIHIRO; WATANABE KATSUYA
Applicant: MITSUBISHI ELECTRIC CORP
Classification:
- international: G06F9/45
- european:
Application number: JP20020278301 20020925
Priority number(s): JP20020278301 20020925

Report a data error here

Abstract of **JP2004118367**
<P>PROBLEM TO BE SOLVED: To provide an intermediate code execution device that performs efficient compilation processing without lowering a reaction speed to a key operation, drawing and the like. <P>SOLUTION: Before an application starting request from a user, a static analysis part 1002 computes a ratio of the code size of each method or each processing unit to the total code size of all intermediate codes included in a transfer file, a compilation determination part 1006 selects a method or processing unit with a ratio of or over a constant level as a target of compilation processing, a compiler 1008 compiles the selected method or processing unit, and a compiled code cache 1014 stores a native code after compilation processing. <P>COPYRIGHT: (C)2004,JPO



Data supplied from the ***esp@cenet*** database - Worldwide

(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-118367
(P2004-118367A)
(43)公開日　平成16年4月15日(2004.4.15)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 9/45	G06F	9/44	322F	5B081
	G06F	9/44	320C	

審査請求 未請求　請求項の数 27　OL　（全 28 頁）

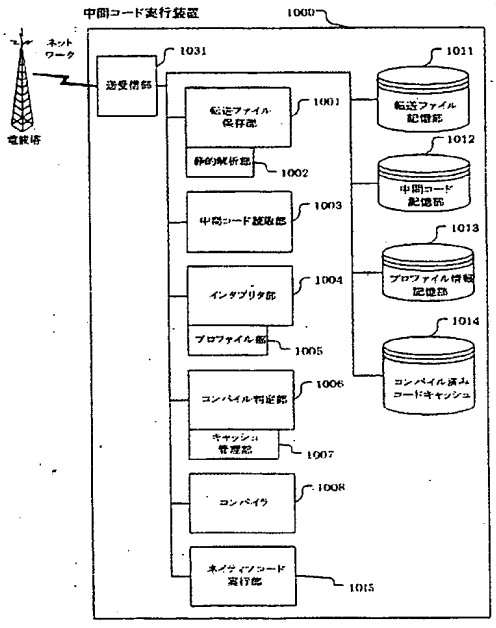
(21)出願番号　特願2002-278301(P2002-278301)
(22)出願日　平成14年9月25日(2002.9.25)

(71)出願人　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目２番３号
(74)代理人　100099461
弁理士　溝井　章司
(74)代理人　100111497
弁理士　波田　啓子
(74)代理人　100111800
弁理士　竹内　三明
(74)代理人　100114878
弁理士　山地　博人
(74)代理人　100118810
弁理士　小原　寿美子

最終頁に続く

(54)【発明の名称】コンピュータ装置及びプログラム実行方法

(57)【要約】
【課題】キー操作、描画等に対する反応速度の低下を伴わずに効率的にコンパイル処理できる中間コード実行装置を提供する。
【解決手段】ユーザからのアプリケーション起動要求前に、事前に、静的解析部１００２が転送ファイルに含まれる全中間コードの総コードサイズに対する各メソッド又は各処理単位のコードサイズの比率を算出し、コンパイル判定部１００６が一定レベル以上の比率を有するメソッド又は処理単位をコンパイル処理の対象として選択し、コンパイラ１００８が選択されたメソッド又は処理単位のコンパイル処理を行い、コンパイル済みコードキャッシュ１０１４がコンパイル処理後のネイティブコードを格納する。
【選択図】　図１



【特許請求の範囲】
【請求項１】
中間コードで表現された中間コードプログラムを実行するコンピュータ装置であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイラと、
部分プログラムごとに、部分プログラムの実行状況に依存しない静特性に関する解析を行うプログラム解析部と、
前記プログラム解析部の解析結果に基づき複数の部分プログラムの中からコンパイル処理の対象とする部分プログラムを選択するコンパイル判定部とを有し、
前記コンパイラは、
前記コンパイル判定部により選択された部分プログラムのコンパイル処理を行うことを特徴とするコンピュータ装置。
【請求項２】
前記プログラム解析部は、
いずれの部分プログラムに関しても実行実績がない状況で、部分プログラムごとに静特性の解析を行い、
前記コンパイル判定部は、
いずれの部分プログラムに関しても実行実績がない状況で、前記プログラム解析部の解析結果に基づきコンパイル処理の対象とする部分プログラムを選択し、
前記コンパイラは、
いずれの部分プログラムに関しても実行実績がない状況で、前記コンパイル判定部により選択された部分プログラムのコンパイル処理を行うことを特徴とする請求項１に記載のコンピュータ装置。
【請求項３】
前記プログラム解析部は、
いずれの部分プログラムも実行されていない状況で、部分プログラムごとに静特性の解析を行い、
前記コンパイル判定部は、
いずれの部分プログラムも実行されていない状況で、前記プログラム解析部の解析結果に基づきコンパイル処理の対象とする部分プログラムを選択し、
前記コンパイラは、
いずれの部分プログラムも実行されていない状況で、前記コンパイル判定部により選択された部分プログラムのコンパイル処理を行うことを特徴とする請求項１に記載のコンピュータ装置。
【請求項４】
前記プログラム解析部は、
いずれかの部分プログラムが実行されている状況で、部分プログラムごとに静特性の解析を行い、
前記コンパイル判定部は、
いずれかの部分プログラムが実行されている状況で、コンパイル処理の対象とする部分プログラムを選択し、
前記コンパイラは、
いずれかの部分プログラムが実行されている状況で、前記コンパイル判定部により選択された部分プログラムのコンパイル処理を行うことを特徴とする請求項１に記載のコンピュータ装置。
【請求項５】
前記プログラム解析部は、
静特性の解析として、それぞれの部分プログラムのコードサイズを解析するとともに中間コードプログラムの総コードサイズに対するそれぞれの部分プログラムのコードサイズの比率を算出し、

前記コンパイル判定部は、
複数の部分プログラムの中から一定レベル以上の比率を有する部分プログラムをコンパイル処理の対象として選択することを特徴とする請求項１～４のいずれかに記載のコンピュータ装置。
【請求項６】
前記プログラム解析部は、
静特性の解析として、そのぞれの部分プログラムの機能を解析し、
前記コンパイル判定部は、
複数の部分プログラムの中から特定の機能を有する部分プログラムをコンパイル処理の対象として選択することを特徴とする請求項１～４のいずれかに記載のコンピュータ装置。
【請求項７】
前記プログラム解析部は、
静特性の解析として、複数の部分プログラムのうちの少なくとも一つ以上の特定の部分プログラムの機能を解析し、
前記コンパイル判定部は、
前記プログラム解析部による解析の対象とされた部分プログラムの中から特定の機能を有する部分プログラムをコンパイル処理の対象として選択することを特徴とする請求項１～４のいずれかに記載のコンピュータ装置。
【請求項８】
前記コンパイル判定部は、
コンパイル処理の対象として特定の部分プログラムを選択した場合に、選択した部分プログラムに関連する部分プログラムが存在するか否かを判断し、関連する部分プログラムが存在する場合に当該関連する部分プログラムをコンパイル処理の対象として選択することを特徴とする請求項１に記載のコンピュータ装置。
【請求項９】
前記コンパイル判定部は、
前記プログラム解析部の解析結果に基づき複数の部分プログラムの中からコンパイル処理の対象としない部分プログラムを選択し、
前記コンパイラは、
前記コンパイル判定部により選択された部分プログラムに対してはコンパイル処理を行わないことを特徴とする請求項１に記載のコンピュータ装置。
【請求項１０】
中間コードで表現された中間コードプログラムを実行するコンピュータ装置であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイラと、
部分プログラムの実行状況をモニターし、それぞれの部分プログラムの実行状況を示すプロファイル情報を作成するプロファイル部と、
複数の部分プログラムの中からコンパイル処理の対象とする部分プログラムを選択するコンパイル判定部とを有し、
前記コンパイラは、
前記プロファイル部により作成されたプロファイル情報に基づき、前記コンパイル判定部によりコンパイル処理の対象として選択された部分プログラムについてコンパイル処理の順序を決定し、決定したコンパイル処理の順序に従って部分プログラムのコンパイル処理を行うことを特徴とすることを特徴とするコンピュータ装置。
【請求項１１】
中間コードで表現された中間コードプログラムを実行するコンピュータ装置であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイラと、
部分プログラムごとに、部分プログラムの実行状況に依存しない静特性に関する解析を行うプログラム解析部と、

複数の部分プログラムの中からコンパイル処理の対象とする部分プログラムを選択するコンパイル判定部とを有し、
前記コンパイラは、
前記プログラム解析部の解析結果に基づき、前記コンパイル判定部によりコンパイル処理の対象として選択された部分プログラムについてコンパイル処理の順序を決定し、決定したコンパイル処理の順序に従って部分プログラムのコンパイル処理を行うことを特徴とすることを特徴とするコンピュータ装置。
【請求項１２】
前記プログラム解析部は、
静特性の解析として、それぞれの部分プログラムのコードサイズを解析するとともに中間コードプログラムの総コードサイズに対するそれぞれの部分プログラムのコードサイズの比率を算出し、
前記コンパイラは、
コンパイル処理の対象として選択された部分プログラムのうち一定レベル以上の比率を有する部分プログラムを優先してコンパイル処理の順序を決定することを特徴とする請求項１１に記載のコンピュータ装置。
【請求項１３】
前記コンパイラは、
一定レベル以上の比率を有する部分プログラムに関連する部分プログラムがコンパイル処理の対象として選択されているか否かを判断し、関連する部分プログラムがコンパイル処理の対象として選択されている場合に当該関連する部分プログラムを優先してコンパイル処理の順序を決定することを特徴とする請求項１２に記載のコンピュータ装置。
【請求項１４】
前記プログラム解析部は、
静特性の解析として、そのぞれの部分プログラムの機能を解析し、
前記コンパイラは、
コンパイル処理の対象として選択された部分プログラムのうち特定の機能を有する部分プログラムを優先してコンパイル処理の順序を決定することを特徴とする請求項１１に記載のコンピュータ装置。
【請求項１５】
前記コンパイラは、
特定の機能を有する部分プログラムに関連する部分プログラムがコンパイル処理の対象として選択されているか否かを判断し、関連する部分プログラムがコンパイル処理の対象として選択されている場合に当該関連する部分プログラムを優先してコンパイル処理の順序を決定することを特徴とする請求項１４に記載のコンピュータ装置。
【請求項１６】
前記コンパイル判定部は、
コンパイル処理の対象として選択した部分プログラムをコンパイル処理キューに設定し、
前記コンパイラは、
決定したコンパイル処理の順序に従って、前記コンパイル判定部によりコンパイル処理キューに設定された部分プログラムの並べ替えを行うことを特徴とする請求項１０又は１１に記載のコンピュータ装置。
【請求項１７】
中間コードで表現された中間コードプログラムを実行するコンピュータ装置であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイラと、
前記コンパイラによりコンパイル処理された部分プログラムを格納するコンパイル処理後プログラム格納部と、
部分プログラムの実行状況をモニターし、それぞれの部分プログラムの実行状況を示すプロファイル情報を作成するプロファイル部と、

前記プロファイル部により作成されたプロファイル情報に基づき前記コンパイル処理後プログラム格納部に格納されている部分プログラムの中から特定の部分プログラムを選択し、選択した部分プログラムを前記コンパイル処理後プログラム格納部より削除するプログラム管理部とを有することを特徴とするコンピュータ装置。

【請求項１８】

中間コードで表現された中間コードプログラムを実行するコンピュータ装置であって、

中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイラと、

前記コンパイラによりコンパイル処理された部分プログラムを格納するコンパイル処理後プログラム格納部と、

部分プログラムごとに、部分プログラムの実行状況に依存しない静特性に関する解析を行うプログラム解析部と、

前記プログラム解析部の解析結果に基づき前記コンパイル処理後プログラム格納部に格納されている部分プログラムの中から特定の部分プログラムを選択し、選択した部分プログラムを前記コンパイル処理後プログラム格納部より削除するプログラム管理部とを有することを特徴とするコンピュータ装置。

【請求項１９】

前記プログラム解析部は、

静特性の解析として、それぞれの部分プログラムのコードサイズを解析するとともに中間コードプログラムの総コードサイズに対するそれぞれの部分プログラムのコードサイズの比率を算出し、

前記プログラム管理部は、

前記コンパイル処理後プログラム格納部に格納されている部分プログラムのうち一定レベル以上の比率を有する部分プログラムを削除対象として選択しないことを特徴とする請求項１８に記載のコンピュータ装置。

【請求項２０】

前記プログラム管理部は、

一定レベル以上の比率を有する部分プログラムに関連する部分プログラムが前記コンパイル処理後プログラム格納部に格納されているか否かを判断し、関連する部分プログラムが格納されている場合に当該関連する部分プログラムを削除対象として選択しないことを特徴とする請求項１９に記載のコンピュータ装置。

【請求項２１】

前記プログラム解析部は、

静特性の解析として、そのぞれの部分プログラムの機能を解析し、

前記プログラム管理部は、

前記コンパイル処理後プログラム格納部に格納されている部分プログラムのうち特定の機能を有する部分プログラムを削除対象として選択しないことを特徴とする請求項２０に記載のコンピュータ装置。

【請求項２２】

前記プログラム管理部は、

特定の機能を有する部分プログラムに関連する部分プログラムが前記コンパイル処理後プログラム格納部に格納されているか否かを判断し、関連する部分プログラムが格納されている場合に当該関連する部分プログラムを削除対象として選択しないことを特徴とする請求項２１に記載のコンピュータ装置。

【請求項２３】

中間コードで表現された中間コードプログラムを実行するプログラム実行方法であって、

中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイル処理ステップと、

部分プログラムごとに、部分プログラムの実行状況に依存しない静特性に関する解析を行うプログラム解析ステップと、

前記プログラム解析ステップでの解析結果に基づき複数の部分プログラムの中からコンパイル処理の対象とする部分プログラムを選択するコンパイル判定ステップとを有し、
前記コンパイル処理ステップは、
前記コンパイル判定ステップで選択された部分プログラムのコンパイル処理を行うことを特徴とするプログラム実行方法。
【請求項２４】
中間コードで表現された中間コードプログラムを実行するプログラム実行方法であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイル処理ステップと、
部分プログラムの実行状況をモニターし、それぞれの部分プログラムの実行状況を示すプロファイル情報を作成するプロファイル情報作成ステップと、
複数の部分プログラムの中からコンパイル処理の対象とする部分プログラムを選択するコンパイル判定ステップとを有し、
前記コンパイル処理ステップは、
前記プロファイル情報作成ステップで作成されたプロファイル情報に基づき、前記コンパイル判定ステップでコンパイル処理の対象として選択された部分プログラムについてコンパイル処理の順序を決定し、決定したコンパイル処理の順序に従って部分プログラムのコンパイル処理を行うことを特徴とすることを特徴とするプログラム実行方法。
【請求項２５】
中間コードで表現された中間コードプログラムを実行するプログラム実行方法であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイル処理ステップと、
部分プログラムごとに、部分プログラムの実行状況に依存しない静特性に関する解析を行うプログラム解析ステップと、
複数の部分プログラムの中からコンパイル処理の対象とする部分プログラムを選択するコンパイル判定ステップとを有し、
前記コンパイル処理ステップは、
前記プログラム解析ステップでの解析結果に基づき、前記コンパイル判定ステップでコンパイル処理の対象として選択された部分プログラムについてコンパイル処理の順序を決定し、決定したコンパイル処理の順序に従って部分プログラムのコンパイル処理を行うことを特徴とすることを特徴とするプログラム実行方法。
【請求項２６】
中間コードで表現された中間コードプログラムを実行するプログラム実行方法であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイル処理ステップと、
前記コンパイル処理ステップによりコンパイル処理された部分プログラムを格納するコンパイル処理後プログラム格納ステップと、
部分プログラムの実行状況をモニターし、それぞれの部分プログラムの実行状況を示すプロファイル情報を作成するプロファイル情報作成ステップと、
前記プロファイル情報作成ステップで作成されたプロファイル情報に基づき前記コンパイル処理後プログラム格納ステップにより格納された部分プログラムの中から特定の部分プログラムを選択し、選択した部分プログラムを削除するプログラム管理ステップとを有することを特徴とするプログラム実行方法。
【請求項２７】
中間コードで表現された中間コードプログラムを実行するプログラム実行方法であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイル処理ステップと、
前記コンパイル処理ステップによりコンパイル処理された部分プログラムを格納するコンパイル処理後プログラム格納ステップと、
部分プログラムごとに、部分プログラムの実行状況に依存しない静特性に関する解析を行

うプログラム解析ステップと、
前記プログラム解析ステップでの解析結果に基づき前記コンパイル処理後プログラム格納ステップにより格納された部分プログラムの中から特定の部分プログラムを選択し、選択した部分プログラムを削除するプログラム管理ステップとを有することを特徴とするプログラム実行方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、Ｊａｖａ（登録商標）やＳｍａｌｌＴａｌｋ等の中間コードであるバイトコードをインタプリタにて実行する場合に動的にコンパイラを使用する技術に関して、コンパイルを行うかの判定技術、コンパイル対象の優先度の管理技術、コンパイル済みコードのキャッシュ管理技術に関する。
【０００２】
【従来の技術】
特開平１１－２７２４７６公報に開示の「バイトコードされたプログラムを動的に最適化するための方法及び装置」では、コンパイル対象となる中間コードのある単位（メソッド、ループ内処理）に対して、最近の一定時間内に呼ばれた回数、累積時間報、コンパイル処理時のオーバヘッド予測、インタプリタのオーバヘッド時間等のプロファイル情報を用いてコンパイルを行うか判断することで、全体の性能向上に対する効果を上げることができる。
【０００３】
特開２０００－０４０００７公報に開示の「バイトコード・コンパイラのためのコード生成」では、インライン展開、仮想関数の解決手段等のコンパイル済みコードを高速に実行する仕組みと共に、従来のバイトコード・インタプリタと並行してコンパイル処理を実現し、コンパイル済みコードはキャッシュを用いて管理することで、コンパイル処理のオーバヘッドを低減し、全体の性能向上に対する効果を上げることができる。
【０００４】
【特許文献１】
特開平１１－２７２４７６
【特許文献２】
特開２０００－０４０００７
【０００５】
【発明が解決しようとする課題】
しかしながら、特開平１１－２７２４７６の方法においては、コンパイル処理のオーバヘッドの時間は、アプリケーションの利用者のキー操作、描画等の反応速度を低下させて、アプリケーションの動作が一瞬止まるような現象を引き起こす。特開２０００－０４０００７の方法においては、並行して行われるコンパイル処理の優先度を、非常に低く設定することで上記のアプリケーション動作が一瞬止まる現象を引き起こさない。しかし、並行して行われるコンパイル処理の優先度が低く、コンパイル処理に対するキューが処理されず、性能向上に結びつかない。
【０００６】
本発明の目的は、コンパイル処理のオーバヘッドによるキー操作、描画等の反応速度の低下をなくし、性能が向上する中間コードのコンパイラを使用する方法及び装置を提供する。
【０００７】
【課題を解決するための手段】
本発明に係るコンピュータ装置は、
中間コードで表現された中間コードプログラムを実行するコンピュータ装置であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイラと、

部分プログラムごとに、部分プログラムの実行状況に依存しない静特性に関する解析を行うプログラム解析部と、
前記プログラム解析部の解析結果に基づき複数の部分プログラムの中からコンパイル処理の対象とする部分プログラムを選択するコンパイル判定部とを有し、
前記コンパイラは、
前記コンパイル判定部により選択された部分プログラムのコンパイル処理を行うことを特徴とする。
【０００８】
本発明に係るコンピュータ装置は、
中間コードで表現された中間コードプログラムを実行するコンピュータ装置であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイラと、
部分プログラムの実行状況をモニターし、それぞれの部分プログラムの実行状況を示すプロファイル情報を作成するプロファイル部と、
複数の部分プログラムの中からコンパイル処理の対象とする部分プログラムを選択するコンパイル判定部とを有し、
前記コンパイラは、
前記プロファイル部により作成されたプロファイル情報に基づき、前記コンパイル判定部によりコンパイル処理の対象として選択された部分プログラムについてコンパイル処理の順序を決定し、決定したコンパイル処理の順序に従って部分プログラムのコンパイル処理を行うことを特徴とすることを特徴とする。
【０００９】
本発明に係るコンピュータ装置は、
中間コードで表現された中間コードプログラムを実行するコンピュータ装置であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイラと、
部分プログラムごとに、部分プログラムの実行状況に依存しない静特性に関する解析を行うプログラム解析部と、
複数の部分プログラムの中からコンパイル処理の対象とする部分プログラムを選択するコンパイル判定部とを有し、
前記コンパイラは、
前記プログラム解析部の解析結果に基づき、前記コンパイル判定部によりコンパイル処理の対象として選択された部分プログラムについてコンパイル処理の順序を決定し、決定したコンパイル処理の順序に従って部分プログラムのコンパイル処理を行うことを特徴とすることを特徴とする。
【００１０】
本発明に係るコンピュータ装置は、
中間コードで表現された中間コードプログラムを実行するコンピュータ装置であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイラと、
前記コンパイラによりコンパイル処理された部分プログラムを格納するコンパイル処理後プログラム格納部と、
部分プログラムの実行状況をモニターし、それぞれの部分プログラムの実行状況を示すプロファイル情報を作成するプロファイル部と、
前記プロファイル部により作成されたプロファイル情報に基づき前記コンパイル処理後プログラム格納部に格納されている部分プログラムの中から特定の部分プログラムを選択し、選択した部分プログラムを前記コンパイル処理後プログラム格納部より削除するプログラム管理部とを有することを特徴とする。
【００１１】
本発明に係るコンピュータ装置は、

中間コードで表現された中間コードプログラムを実行するコンピュータ装置であって、
中間コードプログラムを構成する複数の部分プログラムについて、部分プログラムごとのコンパイル処理が可能なコンパイラと、
前記コンパイラによりコンパイル処理された部分プログラムを格納するコンパイル処理後プログラム格納部と、
部分プログラムごとに、部分プログラムの実行状況に依存しない静特性に関する解析を行うプログラム解析部と、
前記プログラム解析部の解析結果に基づき前記コンパイル処理後プログラム格納部に格納されている部分プログラムの中から特定の部分プログラムを選択し、選択した部分プログラムを前記コンパイル処理後プログラム格納部より削除するプログラム管理部とを有することを特徴とする。

【００１２】

【発明の実施の形態】

実施の形態１.

本実施の形態に関わる中間コードプログラムを実行する装置について説明する。
中間コードプログラムは、中間コードで表現されたプログラムであり、メソッドや処理単位（ＦＯＲ文、ＩＦ文等のプログラム実行時に実行単位となるコードブロック）といった部分プログラムにより構成される。以下、中間コードプログラムを中間コードと記す。

【００１３】

図１は、本実施の形態に係る中間コードを実行する中間コード実行装置１０００の構成図である。中間コード実行装置１０００は、ＣＰＵ、メモリ、記憶装置、通信制御装置、入出力装置等を備えるコンピュータで構成される。

【００１４】

中間コード実行装置１０００を操作するユーザの要求により、送受信部１０３１は、中間コードにより構成する複数のクラスファイルを含むプログラム及びプログラムが使用する画像、音声等のリソースデータを含む転送ファイルを、ネットワークから受信する。
転送ファイル保存部１００１は、転送ファイルを転送ファイル記憶部１０１１に保存する。

【００１５】

中間コード実行装置１０００を操作するユーザの要求により、転送ファイル記憶部１０１１に保存された転送ファイル内に含まれる各クラスファイル及びクラスファイル内の中間コードを、中間コード読取部１００３により読み込み、インタプリタ部１００４が解釈できる記憶形式として中間コード記憶部１０１２に格納する。
中間コード記憶部１０１２に格納される記憶形式には、クラス名、クラスが継承する親クラス、クラス内のメソッドの数等のクラスの属性情報、メソッド名、クラス内のフィールド、メソッド内のフィールドの数及びその名前の属性情報、各メソッドの中間コードが含まれている。

【００１６】

そして、中間コード実行装置１０００のインタプリタ部１００４は、中間コード記憶部１０１２に格納した中間コードを逐次解釈して実行する。中間コードとして記述されたメソッド呼出の命令では、呼び出すべきクラス及びメソッドが指定されており、インタプリタ部１００４は、中間コード記憶部１０１２に格納されたクラス及びメソッドの属性情報を解釈してメソッドを呼び出す。また、中間コードとして記述されたフィールド参照では、参照すべきクラス及びメソッドが指定されており、インタプリタ部１００４は、中間コード記憶部１０１２に格納されたクラス及びメソッドの属性情報を解釈してフィールドを参照する。

【００１７】

また、インタプリタ部１００４は、中間コードのプログラム実行と共に、プロファイル部１００５を制御して、プロファイル情報を取得する。
プロファイル部１００５は、各メソッド、各処理単位等の実行回数及び実行時間といった

実行状況をモニターし、実行状況を示すプロファイル情報を生成し、生成したプロファイル情報をプロファイル情報記憶部１０１３に格納する。

【００１８】

中間コード実行装置１０００のコンパイル判定部１００６は、静的解析部１００２による静的解析結果に基づき、メソッド又は処理単位ごとにコンパイラによるコンパイル処理の対象とするか否かの判断を行う。静的解析部１００２は、メソッドまたは処理単位ごとに、メソッドまたは処理単位の実行状況に依存しない静特性について解析を行う。静的解析部１００２はプログラム解析部の例に相当する。

コンパイル判定部１００６がコンパイル処理の対象と判断した場合には、コンパイラ１００８が、メソッド又は処理単位ごとにネイティブコード実行部１０１５が直接実行できるネイティブコードを生成し、コンパイル済みコードキャッシュ１０１４に格納する。格納されたコンパイル済みコードキャッシュ１０１４は、ネイティブコード実行部１０１５により実行される。

【００１９】

また、コンパイル判定部１００６は、キャッシュ管理部１００７を制御して、コンパイル済みコードキャッシュ１０１４に、コンパイル対象であるメソッド又は処理単位がコンパイルされているネイティブコードが存在するか調べる。コンパイル済みコードが存在すればコンパイラ１００８はコンパイルせずに、ネイティブコード実行部１０１５がコンパイル済みコードを実行する。また、コンパイル済みコードが存在しない場合には、インタプリタ部１００４が該当メソッドまたは処理単位を実行する。

また、コンパイル済みコードキャッシュ１０１４は、コンパイル処理後プログラム格納部の例に相当する。

【００２０】

さらに、キャッシュ管理部１００７は、コンパイル済みコードキャッシュ１０１４に、新たにコンパイルしたコンパイル済みコードを格納する領域がない場合には、コンパイル済みコードキャッシュに含まれるコンパイル済みコードの最終呼出時間等を用いて、例えば呼出が最も古いコンパイル済みコードを選択し、コンパイル済みコードキャッシュ１０１４から選択したコンパイル済みコードを削除する。

キャッシュ管理部１００７は、プログラム管理部の例に相当する。

【００２１】

本実施の形態では、転送ファイル保存部１００１は、転送ファイル記憶部１０１１に転送ファイルを保存するだけではなく、中間コード読取部１００３を制御する。中間コード読取部１００３は、転送ファイル記憶部１０１１に保存された転送ファイルに含まれる複数のクラスファイル及びクラスファイル内の中間コードを読み取り、中間コード記憶部１０１２に格納する。また、転送ファイル保存部１００１は、静的解析部１００２を制御して、中間コード記憶部１０１２に格納された中間コードの各メソッド、各処理単位について静的特性の解析を行わせる。

【００２２】

また、転送ファイル保存部１００１は、コンパイル判定部を制御し、静的解析部１００２の解析結果の内容を判断してコンパイルを行うべきメソッド、処理単位を見つけ出させ、またコンパイラ１００８に該当するメソッド、処理単位についてコンパイルを行わせる。コンパイラ１００８は、コンパイル済みコードをコンパイル済みコードキャッシュ１０１４に格納する。

【００２３】

次に、静的解析部１００２が行う静的解析及び静的解析の結果を用いるコンパイル判定部の詳細な動作について示す。

【００２４】

図２は、中間コードを用いるゲームプログラムの各メソッドのコードサイズ及び転送ファイル内の全中間コードの総コードサイズに対する割合、それに加えて、実行時の呼出回数及び全呼出数に対する各メソッドの呼出回数の割合を記載している。

【００２５】
図２のＧａｍｅＣｌａｓｓ＄ＣｌａｓｓＡクラスのＭｅｈｔｈｏｄＡ（）Ｖに含まれるコードサイズは、転送ファイル内の全中間コードの総コードサイズに対する４７％である。また、全体に対する割合が多いことは、アプリケーション全体に対して多くの処理のための中間コードが含まれていることを示しており、ＧａｍｅＣｌａｓｓ＄ＣｌａｓｓＡクラスのＭｅｈｔｈｏｄＡ（）Ｖの呼出回数は、全体の呼出回数に対して高い割合（図２では６７％）となる。
図２に示したような転送ファイルに含まれる全中間コードの総コードサイズのうちの多くが一つのメソッドに含まれるアプリケーションは、下記の理由により多く存在すると考えられる。

【００２６】
特開２０００－４０００７公報に開示の「バイトコード・コンパイラのためのコード生成」では、呼出先メソッドを呼出元メソッド内の処理としてインライン展開し、ネイティブコードをキャッシュとして管理する技術が示されており、インライン展開が性能向上に大きく貢献することが記載されている。しかしながら、中間コードのアプリケーションは、様々な仮想マシン、中間コード実行装置において実行されるため、仮想マシンに特開２０００－４０００７が示す機能を期待せずに作成されるために、最初からインライン展開を行ったメソッドを作成すると考えられる。また、特開２０００－４０００７では、インライン展開のためのコンパイル処理のオーバヘッド処理が含まれるために、コンパイル及びインライン展開の処理が行われることによるユーザ操作に対する反応速度の低下や画面描画の一時的な停止を招く恐れがある。

【００２７】
以上の二つの理由から、中間コードを用いたアプリケーションの最もよく利用される中間コード群は、最初からインライン展開を行ったクラス内の一つのメソッドとして提供されると考えられる。つまり、最もよく実行される中間コード群は、転送ファイルに含まれる全中間コードのコードサイズと各メソッドのコードサイズにより判定することができる。

【００２８】
なお、図２は、各メソッドのコードサイズが全中間コードのコードサイズに占める比率と呼出比率との相関を説明しており、静的解析部１００２の静的解析の結果は、図２の（Ｂ）”転送ファイル内の全中間コードに対する割合”の欄に示すものとなる。

【００２９】
静的解析部１００２は、中間コード記憶部１０１２に格納された転送ファイル内の各クラスの中間コードを積算して全中間コードの総コードサイズを算出し、各メソッド又は各処理単位のコードサイズを取得する。コンパイル判定部は、全中間コードの総コードサイズに対して各メソッド又は各処理単位のコードサイズがある一定の割合、例えば３０％以上であれば、コンパイルを行うことを判定し、コンパイラによりＧａｍｅＣｌａｓｓ＄ＣｌａｓｓＡクラスのＭｅｔｈｏｄＡ（）Ｖは、ネイティブコードに変換されて、ネイティブコードに変換されたコンパイル済みコードは、コンパイル済みコードキャッシュ１０１４に格納される。

【００３０】
次に、転送ファイル保存部１００１の制御による静的解析部１００２を用いた場合の転送ファイルの保存から転送ファイルのアプリケーションが実行される処理の流れを示す。

【００３１】
図３は、転送ファイル保存時の処理である。
ステップＳ３００１において、ユーザからの転送ファイルの取得要求を受ける。次に、ステップＳ３００２において、送受信部１０３１がネットワークから転送ファイルを取得する。次に、ステップＳ３００３において、転送ファイル保存部１００１が転送ファイルを転送ファイル記憶部１０１１に保存する。

【００３２】
次に、ステップＳ３００４において、静的解析部１００２が中間コード読取部１００３を

制御して、転送ファイルのクラス及びそのメソッドの属性及び中間コードを読取り、中間コード記憶部１０１２に保存し、静的解析部１００２は、中間コード記憶部１０１２に保存された各メソッド、各処理単位のコードサイズを測定し、転送ファイルに含まれる全中間コードの総コードサイズを取得する。そして、各メソッド、各処理単位について全中間コードの総コードサイズに対する比率を求める。

【００３３】

次に、ステップＳ３００５において、静的解析部１００２がコンパイル判定部１００６を制御して、全中間コードの総コードサイズに対する比率が一定の割合、例えば３０％以上のメソッド又は処理単位が存在すれば当該メソッド又は処理単位のコンパイルを行うことを判定し、ステップＳ３００６に進む。

ステップＳ３００６では、コンパイラ１００８がコンパイル対象のメソッド又は処理単位のネイティブコードを生成する。

ステップＳ３００７で、コンパイラ１００８がコンパイル済みコードキャッシュ１０１４にコンパイル済みコードを格納する。

【００３４】

図４は、転送ファイルのアプリケーションを実行する際の処理である。ステップＳ３１０１において、ユーザからのアプリケーションの起動要求を受ける。

【００３５】

次に、ステップＳ３１０２において、コンパイル判定部１００６は、実行対象のメソッド又は処理対象についてコンパイル済みコードがコンパイル済みコードキャッシュ１０１４に含まれるか確認を行う。ステップＳ３１０３において、実行対象のメソッド又は処理単位についてのコンパイル済みコードがコンパイル済みコードキャッシュ１０１４に含まれている場合にはステップＳ３１０５に進み、ネイティブコード実行部１０１５が格納されているコンパイル済みコードの実行を行う。実行中のコンパイル済みコード内にメソッド呼出の命令が存在した場合には、ステップＳ３１０６に進む。

【００３６】

ステップＳ３１０３において、実行対象のメソッド又は処理単位についてのコンパイル済みコードがコンパイル済みコードキャッシュ１０１４に含まれていない場合にはステップＳ３１０４に進み、インタプリタ部１００４が、中間コード記憶部１０１２に格納された中間コードを実行する。実行中の中間コード内にメソッド呼出の命令が存在した場合には、ステップＳ３１０６に進む。

【００３７】

ステップＳ３１０６において、ユーザからの終了要求又はアプリケーション自身からの終了要求が存在するか確認する。もし、存在した場合には終了する。終了要求が存在しない場合には、ステップＳ３１０２に進み、ステップＳ３１０３以降の処理を繰り返す。

【００３８】

以上の全体動作を示したように、転送ファイル保存部１００１が転送ファイルの保存時に、静的解析部１００２と中間コード読取部１００３とコンパイル判定部１００６を制御して、メソッドまたは処理単位のコードサイズの全中間コードの総コードサイズに対する割合の静的な解析を行い、ある閾値によりコンパイルを行うことにより、呼出頻度の多いメソッドを、各メソッド又は各処理単位についての実行実績がなく、また、各メソッド又は各処理単位が実行されていない状況で事前にコンパイルすることが可能である。これにより、コンパイル処理に伴うオーバヘッドがなくなり、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を招くことがなくなる。また、各メソッド、各処理単位のコードサイズという静特性を用いるため、各メソッド、各処理単位の実行状況をモニターする必要が無く、簡単な機器構成によりコンパイル要否の判定を行うことができる。

【００３９】

なお、以上の説明と異なり、いずれかのメソッド又は処理単位が実行されている状況でコードサイズに関する解析を行い、コンパイル対象のメソッド、処理単位を判定するようにしてもよい。

【００４０】
以上の説明では、静的解析部１００２が解析する静特性としてメソッド、処理単位のコードサイズの比率を用いたが、以下にて、メソッド、処理単位の機能を静特性とする場合を説明する。この場合も以上と同様に、コンパイル処理に伴うオーバヘッドを削除することができる。
図２に示した呼出回数を示す表の呼出回数が２番目、３番目であるＧａｍｅＣｌａｓｓ＄ＣｌａｓｓＡクラスのｐａｉｎｔ（Ｌｃｏｍ／ｓｙｓｔｅｍ／ｕｉ／Ｇｒａｐｈｉｃｓ；）Ｖメソッド及びＧａｍｅＣｌａｓｓ＄ＣｌａｓｓＡクラスのｐｒｏｃｅｓｓＥｖｅｎｔ（ＩＩ）Ｖメソッドは、それぞれ、描画処理及びキーボードイベントを処理するためのメソッドである。それぞれ、中間コードの処理系である仮想マシンが提供する描画処理用及びキーボードイベント処理用のクラスを継承している。描画処理及びキーボード処理を頻繁に行うゲームアプリケーションであれば呼出回数が多くなると考えられる。
【００４１】
この二つのメソッドを、プロファイル情報に伴う呼出回数や累積実行時間によるコンパイル判定を行った場合には、ある程度ユーザがアプリケーションを操作した段階でコンパイル処理が行われ、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を引き起こす。
【００４２】
図３のフローチャート中のステップＳ３００４において、静的解析部１００２が各メソッド、各処理単位の機能を解析し、描画処理及びキーボードイベントを処理するためクラスを継承して処理を行うメソッドをコンパイル判定部１００６に示し、コンパイル判定部１００６は、これらをコンパイル処理とする。
このような解析と判断を行うことで、各メソッド又は各処理単位の実行実績がなく、また、いずれのメソッド又は処理単位も実行されていない状況で事前にコンパイル処理を行うことができるために、アプリケーション実行中に描画処理及びキー操作のイベントを処理する中間コードのコンパイル処理が行われる事態を回避し、ユーザ操作に対する反応速度の低下や画面描画の一時停止を引き起こすことなく、全体の性能を向上することが可能となる。
【００４３】
以上に対して、コードサイズが一定の比率以上のメソッド及び処理単位から呼び出されるメソッド及び処理単位、または、描画処理及びキー操作のイベント処理するクラスのメソッド及び処理単位から呼び出されるメソッド及び処理単位もコンパイル対象に加えることで、さらに性能の向上を実現することができる。
コンパイル対象であるコードサイズが一定比率以上のメソッド、処理単位または描画処理及びキー操作のイベント処理するクラスのメソッド、処理単位に含まれるメソッド呼出命令からクラスと呼び出されるメソッドを特定できる。特定されたメソッドをコンパイル対象とすることで、ユーザ操作に対する反応速度の低下や画面描画の一時停止を引き起こすことなく、全体の性能を向上することが可能となる。
なお、以上では、コンパイル対象のメソッド等から呼び出されるメソッド等をコンパイル対象としたが、これに限らず、コンパイル対象のメソッド等に関連するメソッド等であればコンパイル対象とするようにしてもよい。
【００４４】
以上では、コンパイル対象のメソッド、処理単位から呼び出されるメソッド、処理単位もコンパイル対象に加えている。コンパイル済みコードキャッシュ１０１４の上限がなければ、全てのコンパイル対象のメソッド又は処理単位を事前にコンパイルすることができる。しかし、現実的な中間コード実行装置では、コンパイル済みコードキャッシュ１０１４には上限が設定される。そのため、全てのコンパイル対象であるメソッド又は処理単位を事前にコンパイルできない可能性がある。その場合には、コンパイル判定部１００６が行う、メソッド又は処理単位の実行回数及び実行時間等のプロファイル情報から判断するコンパイルを行うかの判定に対して、描画処理及びキー操作のイベントを処理するクラスの

メソッド、処理単位およびその呼び出されるメソッド、処理単位をコンパイル対象としないことで、ユーザ操作に対する反応速度の低下や画面描画の一時停止を引き起こさないようにすることができる。つまり、特定の機能を有するメソッド、処理単位はコンパイル処理の対象としないようにすることで、コンパイル処理の必要性の高いメソッド処理単位のみをコンパイルすることができる。

【００４５】

上記の実施の形態では、転送ファイル保存部１００１が転送ファイルを転送ファイル記憶部１０１１に保存した後に、中間コード読取部１００３及び静的解析部１００２、コンパイル判定部１００６を制御することにより、呼出回数が多くなることが期待されるメソッドの静的解析を行った。しかしながら、転送ファイル保存時にコンパイルを行うことにより、コンパイル済みコードキャッシュ１０１４にコンパイル済みコードを格納する必要がある。そのため、転送ファイルを複数保存する転送ファイル記憶部１０１１に対処する場合には、コンパイル済みコードキャッシュ１０１４がそれぞれの転送ファイル毎に必要となる。

【００４６】

コンパイル済みコードキャッシュ１０１４の記憶領域を節約するために、静的解析部１００２による静的解析を中間コード読取部１００３が行うことで対応する。つまり、この場合には、中間コード読取部１００３がプログラム解析部に相当することになる。

転送ファイル保存時には、図３のステップＳ３００１～Ｓ３００３までの処理を行い（ステップＳ３００４以降は行わない）、ユーザによるアプリケーションの起動要求時には、図４のステップＳ３１０１とステップＳ３１０２の間で中間コード読取部１００３が転送ファイル記憶部１０１１に格納された転送ファイル内のクラスとそのクラスのメソッドの中間コードを読み込み、その際に起動要求のあったメソッド、処理単位の機能について静的解析を行う。また、コンパイル判定部１００６は中間コード読取部１００３の解析結果の内容を判断してコンパイルを行うべきメソッド、処理単位を見つけ出して、コンパイラ１００８を用いて該当するメソッド、処理単位についてコンパイルを行う。コンパイラ１００８は、コンパイル済みコードをコンパイル済みコードキャッシュ１０１４に格納する。その後、図４のステップＳ３１０３以降の処理が行われる。

【００４７】

このように、コンパイル処理時のオーバヘッドによるユーザ操作時の画面の一時停止を引き起こさずに、転送ファイル毎のコンパイル済みコードキャッシュ１０１４を用意する必要がないため、記憶領域を節約することができる。

【００４８】

実施の形態２．

本実施の形態は、中間コードの実行と並行したコンパイル処理について記述する。

【００４９】

本実施の形態では、図１に示した中間コード実行装置１０００の構成要素に加えて、コンパイル判定部１００６がコンパイル処理対象であるメソッド又は処理単位を登録するキューを設置する。図５にコンパイル処理するためのキュー５０００とコンパイル処理データの登録と取り出しの処理を示す。このコンパイル処理を行うためのキュー５０００は、コンパイル判定部１００６により、コンパイルを行うことを判断した場合に、コンパイル処理データ５００１が新たに追加される。キュー５０００に追加されたコンパイル処理データは、先頭にある最も優先度が高いコンパイル処理データ５００２がコンパイラ１００８により取り出されて、登録されたコンパイル処理対象であるメソッド又は処理単位に対してコンパイルが行われる。コンパイラ１００８は、コンパイル済みコードをコンパイル済みコードキャッシュ１０１４に書き込む。

【００５０】

キューに追加されたコンパイル処理データのデータ構造を図６に示す。キューに追加されるコンパイル処理データ６０００には、クラス名やメソッド名や処理単位の範囲などのコンパイル処理対象を特定する内容６００１に加えて、優先度を設定するための情報が含ま

れる。優先度を設定するための情報として、プロファイル部１００５がインタプリタ部１００４の実行するメソッドや処理単位の呼出又は実行回数、累積実行時間等のプロファイル情報６００２、静的解析部１００２が設定するメソッドや処理単位の中間コードのサイズ等の静的解析情報６００３がある。コンパイル処理データに、直接、優先度を設定するための情報を含ませる必要はない、参照するためのメモリ上のポインタやデータベースに対するインデックス番号等でも良い。

【００５１】

コンパイル判定部１００６は、プロファイル情報記憶部１０１３に格納された各メソッド、各処理単位の実行回数及び実行時間等のプロファイル情報から、ある閾値を超えていた場合等にメソッド又は処理単位をコンパイルするか判断を行う。コンパイルを行うことを判断した場合には、コンパイル判定部１００６は、プロファイル部１００５によるプロファイル情報、静的解析部１００２が生成する静的解析結果を参照して、コンパイル処理データ５００１を生成する。生成したコンパイル処理データ５００１をコンパイル処理キュー５０００に設定する。

【００５２】

コンパイラ１００８は、インタプリタ部１００４が中間コードを実行している間であっても、マルチタスク管理を実現しているコンピュータシステムにおいては、ある規則により実行される。例えば、タスクのスケジューリングには、タスク毎の優先度を設定し、それに応じた実行時間を割り当てるタイムシェアリングのスケジューリングが行われる。ユーザのキー操作や画面描画の処理が一時的な停止を起こさないように、コンパイラ１００８が行うコンパイル処理よりも、インタプリタ部１００４の中間コードの実行に対して、優先的に実行時間が割り当てられる必要があり、また、コンパイラ１００８は、ユーザのキー操作によるイベントを直ちに処理できるように、イベント発生時には現在の処理を一時中断する割り込み可能なタスクとして管理される。

【００５３】

コンパイラ１００８は、システムから時間が割り当てられた場合に、処理キュー５０００の先頭に配置されたコンパイル処理データ５００２を取り出して、コンパイル処理データ５００２に含まれるコンパイル対象を特定するための情報６００１を読み込み、中間コード記憶部１０１２に格納された中間コードを読み込んでコンパイルを行い、コンパイル済みのコードをコンパイル済みコードキャッシュ１０１４に書き込む。

【００５４】

プロファイル部１００５は、各メソッド、各処理単位の実行回数及び実行時間を測定し、測定結果を示すプロファイル情報を作成し、作成したプロファイル情報を、プロファイル情報記憶部１０１３に格納又は更新する。また、コンパイル処理データに含まれるプロファイル情報６００２も更新する。尚、プロファイル情報６００２がメモリのポインタ、データベースのインデックス番号であれば、プロファイル部１００５は、プロファイル情報記憶部１０１３に格納又は更新するのみでよい。

【００５５】

本実施の形態では、コンパイラ１００８は、コンパイル処理キュー５０００からコンパイル処理データを取り出す前に、コンパイル処理キューの優先度情報であるプロファイル情報６００２及び静的解析情報６００３のデータを参照し、各コンパイル処理データの優先度を判断してコンパイル処理の順序を決定し、決定した順序に従ってコンパイル処理キュー５０００の並べ替えを行う。コンパイラ１００８は、並べ替えられたコンパイル処理キュー５０００の先頭から優先度の高いコンパイル処理データを取り出して、コンパイルを行う。

【００５６】

次に、コンパイル処理キューに対するコンパイラ１００８が行う、コンパイル処理キューの並べ替えとコンパイル処理の際の流れを示す。

【００５７】

図７は、コンパイル処理キューに対するコンパイル処理の流れ図である。

ステップＳ７００１において、コンパイラ１００８は、処理キュー５０００にコンパイル処理データが含まれているか確認し、ステップＳ７００２に進む。ステップＳ７００２では、処理キューにコンパイル処理データが含まれていない場合には、ステップＳ７００１に戻る。ステップＳ７００２では、処理キューにコンパイル処理データが含まれている場合には、ステップＳ７００３に進む。

【００５８】

ステップＳ７００３では、キュー内の優先度情報を調査してコンパイル処理の順序を決定し、決定した順序に従って、キュー内のコンパイル処理データを並べ替える。

次に、ステップＳ７００４では、コンパイラ１００８は、コンパイル処理データに含まれるコンパイル処理対象を特定するための情報を参照し、中間コード記憶部１０１２に格納された中間コードをコンパイルする。

次に、ステップＳ７００５では、コンパイラ１００８は、コンパイルしたコードをコンパイル済みコードキャッシュ１０１４に格納する。

【００５９】

ステップＳ７００６において、ユーザからの終了要求又はアプリケーション自身からの終了要求が存在するか確認する。もし、存在した場合には終了する。終了要求が存在しない場合には、ステップＳ７００１に進み、キュー内容の確認を行う。

【００６０】

図８は、ステップＳ７００３が行うコンパイル処理キューの並べ替え動作を示す図である。

本実施の形態では、ステップＳ７００３は、（実行回数）×（中間コードのサイズ）の大きさによる並べ替えと実行回数による並べ替えを行う。

コンパイル処理キューの状態８１００は、ステップＳ７００３を行う前の状態である。コンパイル処理キューの状態８２００は、ステップＳ７００３の処理内で（実行回数）×（中間コードのサイズ）の大きさによる並べ替えを行った結果を示している。コンパイル処理キューの状態８３００は、第２の優先度である実行回数により並べ替えた結果である。実行回数による並べ替えでは、隣合うコンパイル処理データの（実行回数）×（中間コードのサイズ）が同じ場合に、実行回数の多くなっているコンパイル処理データを前として入れ替えを行う。

【００６１】

コンパイラ１００８がコンパイル処理前に、コンパイル処理キューのコンパイル処理データの順序を入れ替えることにより、コンパイル処理キューに投入されている間の、インタプリタ部１００４がコンパイル対象であるメソッド又は処理単位の実行回数による動的な挙動を反映させることができる。ユーザのキー操作や描画処理の一時停止を起こさないために、コンパイル処理を行うタスクの優先度を低く設定する必要があり、コンパイル処理キューに多くのコンパイル処理が設定される状態が発生するような状況であっても、最も頻繁に利用されるメソッド又は処理単位を優先的にコンパイルすることができる。そのため、全体の性能向上をもたらすコンパイル処理が行われる。

【００６２】

別の形態として、中間コードのサイズを用いずに、プロファイル情報である累積実行時間により並べ替えることとしてもよい。例えば、コンパイル対象としてメソッドを単位として行うコンパイラであれば、メソッドごとにコードサイズが解析されることになる。しかし、メソッドには、条件分岐により実行されてないコードが含まれており、実行回数と中間コードのサイズの積が、実際のメソッドの実行時間でない場合がある。このため、各メソッドの累積実行時間を利用する場合の方がより精度の高い並べ替えを行うことができる。

【００６３】

別の形態として、転送ファイルに含まれる全中間コードの総コードサイズに対するコンパイル処理の対象であるメソッド、処理単位のコードサイズの割合がある閾値を超えた場合には、（実行回数）×（中間コードのサイズ）の大きさ、又は累積実行時間による判断で

優先度が低くても、優先的に取り扱い、コンパイル処理キューの先頭に配置するとしても良い。図２のＧａｍｅＧａｌａ＄ｃｌａｓＡ：ＭｅｔｈｏｄＡ（）Ｖのように、転送ファイル内の全中間コードの４７％を占めるメソッドでは、コンパイル処理キューに投入された際の実行回数、累積実行時間等が小さな値である場合でも、今後、多くの呼出が行われると予想される。そのため、コンパイル判定部１００６は、転送ファイル内の全中間コードのある一定量、例えば３０％以上のコードサイズを占めるメソッドは、１回目の呼出であっても、また、累積実行時間が短い間であっても、コンパイル処理キューに追加する処理を行う。このコンパイル処理キューへの設定処理に対応して、コンパイラ１００８は、転送ファイル内の全中間コードに対して３０％以上のコードサイズを占めるコンパイル対象メソッドを、優先して取扱い処理キューの先頭に配置する。

【００６４】

このように、転送ファイル内の全中間コードに対する割合が多いメソッドを優先してコンパイル処理を行うことにより、ユーザのキー操作や描画処理の一時停止を起こさないために、コンパイル処理を行うタスクの優先度を低く設定する必要があり、コンパイル処理キューに多くのコンパイル処理データが設定される状態が発生する状況であっても、最も頻繁に利用されるメソッド又は処理単位を優先的にコンパイルすることができる。そのため、全体の性能向上をもたらすコンパイル処理が行われる。また、各メソッド、各処理単位のコードサイズという静特性に基づいてコンパイル処理の順序を決定するため、各メソッド、各処理単位の実行状況をモニターする必要がなく、簡単な機器構成により実現することができる。

【００６５】

別の形態として、特定の機能のクラスやメソッドを優先的に取扱い、コンパイル処理キューの先頭に配置するとしても良い。図２に示した呼出回数を示す表の呼出回数が２番目、３番目であるＧａｍｅＣｌａｓｓ＄ＣｌａｓｓＡクラスのｐａｉｎｔ（Ｌｃｏｍ／ｓｙｓｔｅｍ／ｕｉ／Ｇｒａｐｈｉｃｓ；）Ｖメソッド及びＧａｍｅＣｌａｓｓ＄ＣｌａｓｓＡクラスのｐｒｏｃｅｓｓＥｖｅｎｔ（ＩＩ）Ｖメソッドは、それぞれ、描画処理及びキーボードイベントを処理するためのメソッドである。それぞれ、中間コードの処理系である仮想マシンが提供する描画処理用及びキーボードイベント処理用のクラスを継承している。描画処理及びキーボード処理を頻繁に行うゲームアプリケーションであれば呼出回数が多くなると考えられる。

そのため、コンパイル判定部１００６は、描画処理クラスを継承したクラスのｐａｉｎｔメソッドやキーボードイベントを処理するクラスのｐｒｏｃｅｓｓＥｖｅｎｔメソッドを、１回目の呼出であっても、また、累積実行時間が短い間であっても、コンパイル処理キューに追加する処理を行う。このコンパイル処理キューへの設定処理に対応して、コンパイラ１００８は、描画処理クラスを継承したクラスのｐａｉｎｔメソッドやキーボードイベントを処理するクラスのｐｒｏｃｅｓｓＥｖｅｎｔメソッドを、優先的に取扱い処理キューの先頭に配置する。

【００６６】

このように、特定の機能を有するメソッドを、優先してコンパイル処理を行うことにより、ユーザのキー操作や描画処理の一時停止を起こさないために、コンパイル処理を行うタスクの優先度を低く設定する必要があり、コンパイル処理キューに多くのコンパイル処理データが設定される状態が発生する状況であっても、最も頻繁に利用されるメソッド又は処理単位を優先的にコンパイルすることができる。そのため、全体の性能向上をもたらすコンパイル処理が行われる。

【００６７】

また、以上にて説明した処理により特定のメソッド等を優先的に取扱うだけでなく、優先的に取扱われるメソッド等に関連するメソッド等も優先的に取扱い、当該関連するメソッド等を処理キューの前方に配置するようにしてもよい。関連するメソッド等としては、例えば、優先的に取扱われるメソッド等から呼び出されるメソッド等がある。

【００６８】

実施の形態３．
本実施の形態では、コンパイル済みコードキャッシュの管理について記述する。
【００６９】
図１０の１００００は、コンパイル済みコードキャッシュ（コンパイル処理後プログラム格納部）１０１４に含まれる管理テーブルを示しており、各コンパイル済みコードキャッシュの各コンパイル済みコードに対応した管理情報を複数含んでいる。
図９の９０００は、コンパイル済みコードキャッシュの管理テーブルの各要素の内容を示している。コンパイル済みコードキャッシュの管理テーブルの各要素９０００には、クラス名やメソッド名や処理単位の範囲などのコンパイル処理対象を特定するための情報９００１、コンパイル済みコードの格納位置を示すメモリ番地やコンパイル済みコードのサイズ等のコンパイル済みコードを参照するための情報９００４が含まれている。
【００７０】
コンパイル済みコードキャッシュの管理テーブルの各要素９０００には、上記のコンパイル対象の特定情報９００１及びコンパイル済みコードの参照情報９００４に加えて、コンパイル済みコードキャッシュから追い出されるキャッシュデータの優先度を設定するための情報が含まれる。優先度を設定するための情報として、メソッドや処理単位の呼出又は実行回数、累積実行時間、最終呼出時刻等の実行状況を示すプロファイル情報９００２、静的解析部１００２が設定するメソッドや処理単位のコードサイズの比率等の静的解析情報９００３がある。管理テーブルに、直接、優先度を設定するための情報を含ませる必要はない、参照するためのメモリ上のポインタやデータベースに対するインデックス番号等でも良い。
【００７１】
コンパイラ１００８は、中間コードをコンパイルした後のコンパイル済みコードキャッシュ１０１４に対する書き込み処理時に、コンパイル済みコードキャッシュ１０１４の管理テーブル１００００に、新たな要素を追加して、コンパイル済みコードに関する情報をコンパイル済みコードキャッシュ１０１４内の格納領域に書き込む。
【００７２】
キャッシュ管理部（プログラム管理部）１００７は、コンパイル済みコードキャッシュ１０１４に、新たにコンパイルしたコンパイル済みコードを格納する際に、格納する領域がない場合には、コンパイル済みコードキャッシュ１０１４に含まれるコンパイル済みコードから、特定のコンパイル済みコードを選択して削除する。削除時のコンパイル済みコードの選択には、コンパイル済みコードキャッシュ１０１４の管理テーブルの各要素のプロファイル情報９００２、静的解析情報９００３を用いて、コンパイル済みコードキャッシュから追い出される優先度を求める。
【００７３】
図１１に、キャッシュ管理部１００７が、コンパイル済みコードを格納する流れを示す。キャッシュ管理部１００７は、ステップＳ１１００１において、コンパイル済みコードキャッシュ１０１４の空き容量を取得する。次に、ステップＳ１１００２において、コンパイラ１００８がコンパイルしたコンパイル済みコードのコードサイズとステップＳ１１００１で取得した空き容量を比較する。
【００７４】
コンパイル済みコードのコードサイズが空き容量よりも小さい場合、つまり、コンパイル済みコードがコンパイル済みコードキャッシュ１０１４に格納可能である場合には、ステップＳ１１００３に進む。ステップＳ１１００３では、コンパイル済みコードキャッシュ１０１４の管理テーブルに新たな要素を追加して、コンパイル済みコードを格納する。
【００７５】
コンパイル済みコードのコードサイズが空き容量よりも小さい場合、つまり、コンパイル済みコードがコンパイル済みコードキャッシュ１０１４に格納できない場合には、ステップＳ１１００４に進む。ステップＳ１１００４では、コンパイル済みコードキャッシュ１０１４の管理テーブルの各要素のプロファイル情報９００２、静的解析情報９００３を用

いて、コンパイル済みコードキャッシュ１０１４から追い出されるコンパイル済みコードキャッシュを選択するために、各コンパイル済みコードキャッシュの追い出し判断に用いる優先度を求める。次に、ステップＳ１１００５では、追い出す優先度が最も高い（最も追い出すべき）コンパイル済みコードのデータを削除し、コンパイル済みコードキャッシュ１０１４の管理テーブルの追い出す優先度が最も高い（最も追い出すべき）要素を削除する。次に、ステップＳ１１００１に進む。以上のステップＳ１１００１からＳ１１００５を行うことにより、コンパイラ１００８がコンパイルしたコンパイル済みコードをコンパイル済みコードキャッシュ１０１４に書き込むことができる。

【００７６】

ステップＳ１１００４で行うコンパイル済みコードキャッシュから追い出す優先度の求め方について、図１０のコンパイル済みコードキャッシュの管理テーブル１００００を用いて、コンパイル済みコードキャッシュの一つが選択される処理を記述する。本実施の形態では、今後、最も使用されないものを選択するために、（実行回数）×（中間コードサイズ）、実行回数、最終呼出時間という順序で判断する。コンパイル済みコードキャッシュの管理テーブル１００００に登録された各要素１０００１、１０００２、１０００３の（実行回数）×（中間コードサイズ）の値は、全て１５０００である。次に、実行回数の少ない要素を基に優先度を設定する。実行回数が１００であるｃｌａｓｓＡ：ＭｅｔｈｏｄＢ１０００２、ｃｌａｓｓＡ：ＭｅｔｈｏｄＡ１０００３が該当する。次に、最終呼出時間が古いものを選択する。この結果、ｃｌａｓｓＡ：ＭｅｔｈｏｄＢ１０００２が追い出しの対象として選択される。

本実施の形態では、各判断基準を順番に行うことにより選択をおこなったが、各要素に対して判断基準毎の点数を設定し、各判断基準の点数の合計を求める選択方法も可能である。

【００７７】

以上の全体動作を示したように、コンパイル済みコードキャッシュからの追い出す優先度を、最終呼出時間だけではなく、実行回数及び中間コードサイズ及び実行回数を用いて求めることにより、今後も呼び出される可能性が高いと考えられるメソッド又は処理単位のコンパイル済みコードを、コンパイル済みコードキャッシュから追い出されることを抑制して、再コンパイルの処理の発生頻度を少なくすることができる。これにより、コンパイル処理に伴うオーバヘッドが少なくなり、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を招く恐れが低下する。

【００７８】

別の形態として、コードサイズを用いずに、プロファイル情報である累積実行時間による追い出す優先度を求めても良い。例えば、コンパイル対象としてメソッドを単位として行うコンパイラであれば、メソッドごとにコードサイズが解析されることになる。しかし、メソッドには、条件分岐により実行されてないコードが含まれており、実行回数と中間コードのサイズの積が、実際のメソッドの実行時間でない場合がある。このため、各メソッドの累積実行時間を利用する場合の方がより精度の高い追い出し優先度を求めることができる。

【００７９】

別の形態として、プロファイル情報である実行回数、累積実行時間がある閾値を超えた場合には、コンパイル済みコードキャッシュからの追い出し優先度の計算を行わず、コンパイル済みコードキャッシュから追い出すコンパイル済みコードの選択の範囲外として、コンパイル済みコードキャッシュから削除されないようにしてもよい。

【００８０】

コンパイル判定部１００６が実行頻度又は累積実行時間による判定より、メソッド又は処理単位のコンパイルを行うことを判断し、コンパイラ１００８によりコンパイルが行なわれたコンパイル済みコードは、コンパイル済みコードキャッシュ１０１４に格納される。その後、コンパイル済みコードの呼出が行われ、コンパイル済みコードキャッシュ１０１４の管理テーブルに含まれる実行回数、累積実行時間等のプロファイル情報が更新される

。
【００８１】
コンパイル判定部１００６が実行回数又は累積実行時間による判定より、コンパイル済みコードキャッシュ１０１４に含まれないメソッド又は実行単位のコンパイルを行うことを判断し、コンパイラ１００８がコンパイルを行った後、キャッシュ管理部１００７は、コンパイル済みコードキャッシュ１０１４の管理テーブルを用いて、コンパイしたコードの格納領域の確保を行う。その際に、キャッシュ管理部１００７は、コンパイル済みコードキャッシュ１０１４の管理テーブルに含まれる実行回数及び累積実行時間等がある閾値を超えているメソッド又は処理単位に対して、優先度を求める処理を行わず、コンパイル済みコードキャッシュからの追い出しの対象外とする。
【００８２】
コンパイル済みコードキャッシュの管理テーブルのプロファイル情報に含まれる実行回数、累積時間が多いものは、今後も呼び出される可能性が高いと考えられる。キャッシュ管理部１００７は、該当するメソッド又は処理単位のコンパイル済みコードが、コンパイル済みコードキャッシュから追い出されることを抑制して、再コンパイルの処理の発生頻度を少なくすることができる。これにより、コンパイル処理に伴うオーバヘッドが少なくなり、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を招く恐れが低下する。
【００８３】
また、転送ファイルに含まれる全中間コードの総コードサイズに対して一定比率以上のコードサイズを有するメソッド又は処理単位のコンパイル済みのコードについては、コンパイル済みコードキャッシュからの追い出し優先度の計算を行わず、コンパイル済みコードキャッシュから追い出すコンパイル済みコードの選択の範囲外として、コンパイル済みコードキャッシュから削除されないようにしてもよい。
【００８４】
図２のＧａｍｅＧａｌａ＄ｃｌａｓＡ：ＭｅｔｈｏｄＡ（）Ｖのように、転送ファイル内の全中間コードの総コードサイズに対して４７％の比率を占めるメソッドでは、実行回数、累積実行時間等が小さな値である場合でも、今後、多くの呼出が行われると予想される。そのため、コンパイル判定部１００６は、転送ファイル内の全中間コードのある一定量、例えば３０％以上のコードサイズを占めるメソッドは、１回目の呼出であっても、また、累積実行時間が短い間であっても、コンパイラ１００８は、コンパイル処理を行い、コンパイル済みコードキャッシュ１０１４に格納される。
【００８５】
その後、コンパイル判定部１００６が実行頻度又は累積実行時間による判定より、別のメソッド又は実行単位のコンパイルを行うことを判断し、コンパイルを行った後、キャッシュ管理部１００７は、コンパイル済みコードキャッシュ１０１４の管理テーブルを用いて、コンパイル済みコードの格納領域の確保を行う。その際に、キャッシュ管理部１００７は、転送ファイル内の全中間コードの総コードサイズに対して３０％以上のコードサイズを占めるメソッド又は処理単位に対して追い出しのための優先度を求める処理を行わず、コンパイル済みコードキャッシュからの追い出しの対象外とする。
なお、メソッド、処理単位のコードサイズに関する解析は、実施の形態１と同様に静的解析部が行う。
【００８６】
転送ファイル内の全中間コードに対する割合が多いメソッドは、呼出頻度が多くなると予想される。今後も呼び出される可能性が高いと考えられるメソッド又は処理単位のコンパイル済みコードがコンパイル済みコードキャッシュから追い出されることを抑制して、再コンパイルの処理の発生頻度を少なくすることができる。これにより、コンパイル処理に伴うオーバヘッドが少なくなり、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を招く恐れが低下する。
【００８７】

別の形態として、ある特定の機能のメソッド、ある特定のクラス内のメソッド自身またはその中の処理単位についてのコンパイル済みコードには、コンパイル済みコードキャッシュからの追い出し優先度の計算を行わず、コンパイル済みコードキャッシュから追い出すコンパイル済みコードの選択の範囲外として、コンパイル済みコードキャッシュから削除されないようにしてもよい。

【００８８】

図２に示した呼出回数を示す表の呼出回数が２番目、３番目であるＧａｍｅＣｌａｓｓ＄ＣｌａｓｓＡクラスのｐａｉｎｔ（Ｌｃｏｍ／ｓｙｓｔｅｍ／ｕｉ／Ｇｒａｐｈｉｃｓ；）Ｖメソッド及びＧａｍｅＣｌａｓｓ＄ＣｌａｓｓＡクラスのｐｒｏｃｅｓｓＥｖｅｎｔ（ＩＩ）Ｖメソッドは、それぞれ、描画処理及びキーボードイベントを処理するためのメソッドである。それぞれ、中間コードの処理系である仮想マシンが提供する描画処理用及びキーボードイベント処理用のクラスを継承している。描画処理及びキーボード処理を頻繁に行うゲームアプリケーションであれば呼出回数が多くなると考えられる。また、ユーザがアプリケーションを操作している状態でこの二つのメソッドのコンパイル処理が行われると、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を引き起こす。このため、描画処理やキーボードイベントを処理するといった特定の機能を有するメソッド、処理単位については、コンパイル済みコードキャッシュ１０１４からの追い出しの対象とはしないようにする。

なお、メソッド、処理単位の機能の解析は、実施の形態１と同様に静的解析部が行う。

【００８９】

中間コードの処理系である仮想マシンが提供する描画処理用及びキーボードイベント処理用のクラスを継承しているメソッド、処理単位のように、今後も呼び出される可能性が高いと考えられるメソッド又は処理単位のコンパイル済みコードを、コンパイル済みコードキャッシュから追い出されることを抑制して、再コンパイルの処理を行わないにすることができる。これにより、コンパイル処理に伴うオーバヘッドがなくなり、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を招かない。

【００９０】

また、以上にて説明した処理により特定のコンパイル済みコードを追い出しの対象外とするのみでなく、追い出しの対象外とされたコンパイル済みコードに関連するコンパイル済みコードが存在する場合には、当該関連するコンパイル済みコードも追い出しの対象外としてもよい。関連するコンパイル済みコードとは、例えば、追い出しの対象外とされたコンパイル済みコードから呼び出されるコンパイル済みコードがある。

【００９１】

以上の実施の形態１～３で説明した中間コード実行装置の特徴を以下にて再言する。

【００９２】

実施の形態１に記載の中間コード実行装置は、

中間コードを用いて表現されたプログラムを読み込む中間コード読取部と、読み取られた中間コードを実行するインタプリタ部と、中間コードをネイティブコードとしてコンパイルするコンパイラと、コンパイルしたネイティブコードを一時的に格納するキャッシュ領域を備え、

中間コードを用いたプログラムを解析する（静的）解析部による解析を行うことで、中間コードを用いたプログラムの一部をコンパイルするか判定するコンパイル判定部を有することを特徴とする。

【００９３】

実施の形態１に記載の中間コード実行装置は、

（静的）解析部において求めた、プログラムに含まれる全中間コードに対するコンパイルする処理単位の中間コードの割合により、中間コードを用いたプログラムの一部をコンパイルするか判定するコンパイル判定部を有することを特徴とする。

【００９４】

実施の形態１に記載の中間コード実行装置は、

（静的）解析部において見つけられた特定のコンパイルする処理単位（キーイベント、描画を処理するメソッド等）に対して、コンパイルするか判定するコンパイル判定部を有することを特徴とする。

【００９５】

実施の形態１に記載の中間コード実行装置は、

（静的）解析部において見つけられた特定のコンパイルする処理単位（キーイベント、描画を処理するメソッド等）から呼び出される中間コードの一部を、コンパイルするか判定するコンパイル判定部を有することを特徴とする。

【００９６】

実施の形態１に記載の中間コード実行装置は、

中間コードを含んだプログラムを、中間コードを実行するための装置に取得する際に、（静的）解析を行う解析部を有することを特徴とする。

【００９７】

実施の形態１に記載の中間コード実行装置は、

プログラムから中間コードを読み込む際に、（静的）解析を行う解析部を有することを特徴とする。

【００９８】

実施の形態２に記載の中間コード実行装置は、

中間コードを用いて表現されたプログラムを読み込む中間コード読取部と、読み取られた中間コードを実行するインタプリタ部と、中間コードをネイティブコードとしてコンパイルするコンパイラと、コンパイルしたネイティブコードを一時的に格納するキャッシュ領域を備え、

インタプリタ部による中間コードの実行とコンパイル処理を並行して実行できるように、コンパイル処理を設定する上記のコンパイル判定部と設定されたコンパイル処理を行う上記のコンパイラを備え、

インタプリタ部が実行する中間コードの各部分の実行回数又は累積実行時間等の実行時に測定された情報を用いて、複数のコンパイル処理対象から優先度の高いコンパイル処理対象を選択してコンパイルすることを特徴とする。

【００９９】

実施の形態２に記載の中間コード実行装置は、

複数のコンパイル処理を設定するキューを備えることを特徴とする。

【０１００】

実施の形態２に記載の中間コード実行装置は、

インタプリタ部が実行する中間コードの各部分の実行回数又は累積実行時間等の実行時に測定された情報を用いて、複数のコンパイル処理対象から優先度の高い順序でコンパイル処理対象を並べ変えることを特徴とする。

【０１０１】

実施の形態２に記載の中間コード実行装置は、

中間コードの（静的）解析を行う解析部を有し、（静的）解析部において求めた、プログラムに含まれる全中間コードに対するコンパイルする処理単位の中間コードの割合により、複数のコンパイル処理対象から優先度の高いコンパイル処理対象を選択してコンパイルすることを特徴とする。

【０１０２】

実施の形態２に記載の中間コード実行装置は、

中間コードの（静的）解析を行う解析部を有し、（静的）解析部において求めた、プログラムに含まれる全中間コードに対するコンパイルする処理単位の中間コードの割合により、複数のコンパイル処理対象から優先度の高い順序でコンパイル処理対象を並べ変えることを特徴とする。

【０１０３】

実施の形態２に記載の中間コード実行装置は、

中間コードの（静的）解析を行う解析部を有し、（静的）解析部において見つけられた特定のコンパイルする処理単位（キーイベント、描画を処理するメソッド等）を、複数のコンパイル処理対象から選択してコンパイルすることを特徴とする。

【０１０４】

実施の形態２に記載の中間コード実行装置は、
中間コードの取得の際に（静的）解析を行う解析部を有することを特徴とする。

【０１０５】

実施の形態２に記載の中間コード実行装置は、
プログラムから中間コードを読み込む際に（静的）解析を行う解析部を有することを特徴とする。

【０１０６】

実施の形態３に記載の中間コード実行装置は、
中間コードを用いて表現されたプログラムを読み込む中間コード読取部と、読み取られた中間コードを実行するインタプリタ部と、中間コードをネイティブコードとしてコンパイルするコンパイラと、コンパイルしたネイティブコードを一時的に格納するキャッシュ領域を備え、
新たにコンパイルしたネイティブコードを格納するために、ネイティブコードを格納したキャッシュ領域から、インタプリタ部が実行する中間コードの各部分の実行回数又は累積実行時間等の実行時に測定された情報を用いて、削除するネイティブコードを選択するキャッシュ管理部を備えたことを特徴とする。

【０１０７】

実施の形態３に記載の中間コード実行装置は、
中間コードの（静的）解析を行う解析部を有し、（静的）解析部において求めた、プログラムに含まれる全中間コードに対するコンパイルする処理単位の中間コードの割合により、削除するネイティブコードを選択するキャッシュ管理部を備えたことを特徴とする。

【０１０８】

実施の形態３に記載の中間コード実行装置は、
中間コードの（静的）解析を行う解析部を有し、（静的）解析部において見つけられた特定のコンパイルする処理単位（キーイベント、描画を処理するメソッド等）を、削除するネイティブコードの対象外とするキャッシュ管理部を備えたことを特徴とする。

【０１０９】

実施の形態３に記載の中間コード実行装置は、
中間コードの（静的）解析を行う解析部を有し、（静的）解析部において見つけられた特定のコンパイルする処理単位（キーイベント、描画を処理するメソッド等）から呼び出される中間コードの一部を、削除するネイティブコードの対象外とするキャッシュ管理部を備えたことを特徴とする。

【０１１０】

【発明の効果】

本発明によれば、部分プログラムの実行状況に依存しない静特性に関する解析を行い、静特性の解析結果に基づきコンパイル処理の対象となる部分プログラムを選択してコンパイル処理を行うため、部分プログラムの実行状況をモニターする必要がなく、簡単な機器構成によりコンパイル処理の要否判断を行うことができる。

【０１１１】

また、本発明によれば、部分プログラムが実行されていない状況で事前にコンパイル処理の対象となる部分プログラムを選択してコンパイル処理を行うため、部分プログラムの実行中にコンパイル処理が行われる事態を回避することができ、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を招くことなく効率的なデータ処理が可能となる。

【０１１２】

また、本発明によれば、部分プログラムが実行されている状況でコンパイル処理の対象となる部分プログラムを選択してコンパイル処理を行うため、コンパイル処理済みの部分プ

ログラムを格納するための記憶領域を節約することができる。

【０１１３】

また、本発明によれば、中間コードプログラムの総コードサイズに対するそれぞれの部分プログラムのコードサイズの比率を算出し、一定レベル以上の比率を有する部分プログラムについてコンパイル処理を行うため、呼出頻度の高い部分プログラムのコンパイル処理を事前に行うことができ、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を招くことなく効率的なデータ処理が可能となる。

【０１１４】

また、本発明によれば、それぞれの部分プログラムの機能を解析し、特定の機能を有する部分プログラムについてコンパイル処理を行うため、呼出頻度の高い部分プログラムのコンパイル処理を事前に行うことができ、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を招くことなく効率的なデータ処理が可能となる。

【０１１５】

また、本発明によれば、コンパイル処理の対象として選択された部分プログラムに関連する部分プログラムもコンパイル処理の対象とするため、呼出頻度の高い一群の部分プログラムのコンパイル処理を事前にまとめて行うことができ、ユーザ操作に対する反応速度の低下や画面描画の一時的な停止を招くことなく効率的なデータ処理が可能となる。

【０１１６】

また、本発明によれば、静特性の解析結果に基づきコンパイル処理の対象となる部分プログラムを制限するため、コンパイル処理の必要性が高い部分プログラムのみコンパイル処理を行うことができる。

【０１１７】

また、本発明によれば、部分プログラムの実行状況に基づきコンパイル処理の順序を決定し、決定した順序に従ってコンパイル処理を行うため、頻繁に利用される部分プログラムのコンパイル処理を優先的に行うことができる。

【０１１８】

また、本発明によれば、部分プログラムの実行状況に依存しない静特性に関する解析を行い、静特性の解析結果に基づきコンパイル処理の順序を決定し、決定した順序に従ってコンパイル処理を行うため、頻繁に利用される部分プログラムのコンパイル処理を優先的に行うことができ、また、部分プログラムの実行状況をモニターする必要がないため機器構成を簡単にすることができる。

【０１１９】

また、本発明によれば、優先的に取扱われる部分プログラムに関連する部分プログラムも優先的にコンパイル処理を行うため、頻繁に利用される一群の部分プログラムをまとめて優先的にコンパイル処理することができる。

【０１２０】

また、本発明によれば、コンパイル処理された部分プログラムのうち削除対象となる部分プログラムを部分プログラムの実行状況に基づき選択するため、今後も呼び出される可能性が高いと考えられる部分プログラムが削除される事態を回避し、再コンパイル処理の発生頻度を少なくすることができる。

【０１２１】

また、本発明によれば、部分プログラムの実行状況に依存しない静特性に関する解析を行い、コンパイル処理された部分プログラムのうち削除対象となる部分プログラムを静特性の解析結果に基づき選択するため、今後も呼び出される可能性が高いと考えられる部分プログラムが削除される事態を回避し、再コンパイル処理の発生頻度を少なくすることができ、更に、部分プログラムの実行状況をモニターする必要がないため機器構成を簡単にすることができる。

【０１２２】

また、本発明によれば、削除対象とならない部分プログラムに関連する部分プログラムを削除対象としないことで、今後も呼び出される可能性が高いと考えられる一群の部分プロ

グラムのうちのいずれかが削除される事態を回避し、再コンパイル処理の発生頻度を少なくすることができる。

【図面の簡単な説明】

【図１】中間コード実行装置の構成例を示す図。

【図２】静的解析と呼出割合、呼出回数との関係を説明する図。

【図３】転送ファイル保存時の処理の例を示すフローチャート図。

【図４】アプリケーション実行処理の例を示すフローチャート図。

【図５】コンパイル処理キューの例を示す図。

【図６】コンパイル処理データの構成例を示す図。

【図７】コンパイル処理キューに対する処理の例を示す図。

【図８】コンパイル処理キューの並べ替えを説明する図。

【図９】コンパイル済みコードキャッシュの管理テーブルの要素を示す図。

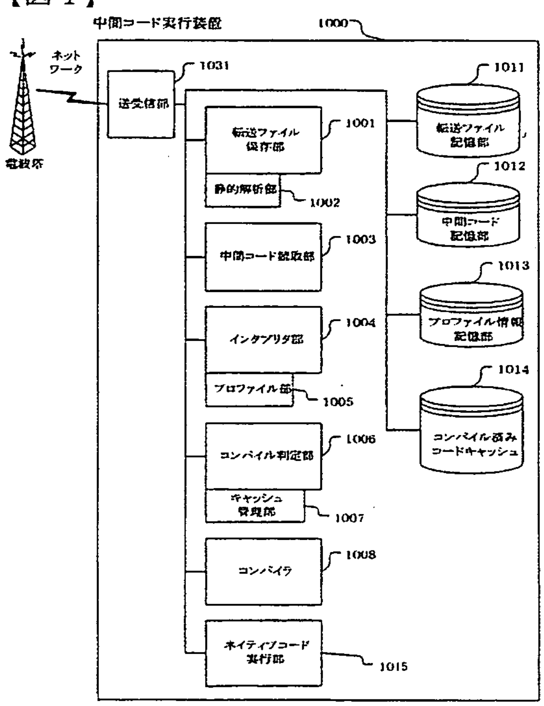
【図１０】コンパイル済みコードキャッシュの管理テーブルの内容例を示す図。

【図１１】コンパイル済みコードの削除処理の例を示すフローチャート図。

【符号の説明】

１０００　中間コード実行装置、１００１　転送ファイル保存部、１００２　静的解析部、１００３　中間コード読取部、１００４　インタプリタ部、１００５　プロファイル部、１００６　コンパイル判定部、１００７　キャッシュ管理部、１００８　コンパイラ、１０１１　転送ファイル記憶部、１０１２　中間コード記憶部、１０１３　プロファイル情報記憶部、１０１４　コンパイル済みコードキャッシュ、１０１５　ネイティブコード実行部、１０３１　送受信部。

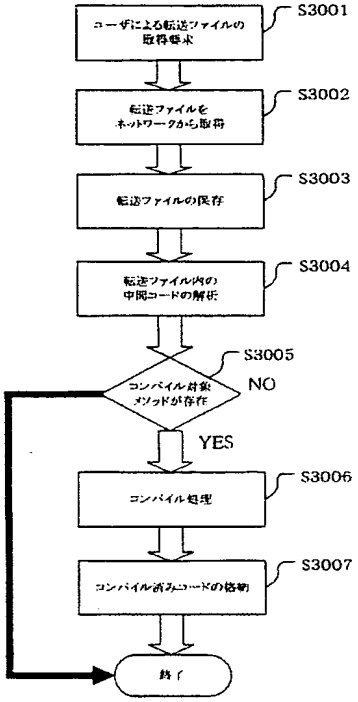
【図１】



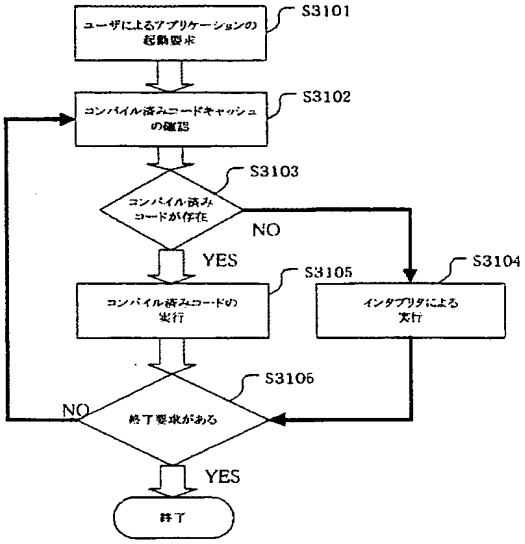
【図２】

メソッド名 クラス名::メソッド名(引数タイプ)返り値タイプ	(A) 各メソッドのコードサイズ(byte)	(B) 転送ファイル内の全中間コードに対する割合	(C) 呼出回数	(D) 呼出割合(%)
GameGala$classA::MethodA()V	3818	47%	813	67%
GameGala$classA::paint(Lcom/system/ui/Graphics;)V	1486	18%	258	21%
GameGala$classA::processEvent(II)V	416	5%	100	8%
GameGala$classA::MethodB()V	58	1%	14	1%
GameGala$classA::MethodC(I)V	988	12%	11	1%
GameGala$classA::MethodD([B)I	47	1%	10	1%
GameGala$classA::MethodE()I	518	6%	5	0%
GameGala $classB::start()V	34	0%	1	0%
GameGala$classA::<init>(LGameGala;)V	731	9%	1	0%
	8094	100%	1211	100

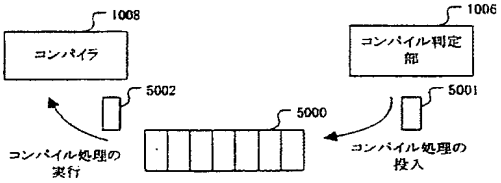
【図３】


ユーザによる転送ファイルの取得要求
S3001
転送ファイルをネットワークから取得
S3002
転送ファイルの保存
S3003
転送ファイル内の中間コードの解析
S3004
コンパイル対象メソッドが存在
S3005
NO
YES
コンパイル処理
S3006
コンパイル済みコードの格納
S3007
終了

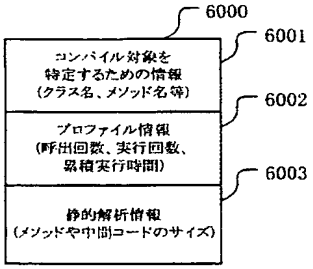
【図４】


ユーザによるアプリケーションの起動要求
S3101
コンパイル済みコードキャッシュの確認
S3102
コンパイル済みコードが存在
S3103
NO
YES
コンパイル済みコードの実行
S3105
インタプリタによる実行
S3104
終了要求がある
S3106
NO
YES
終了

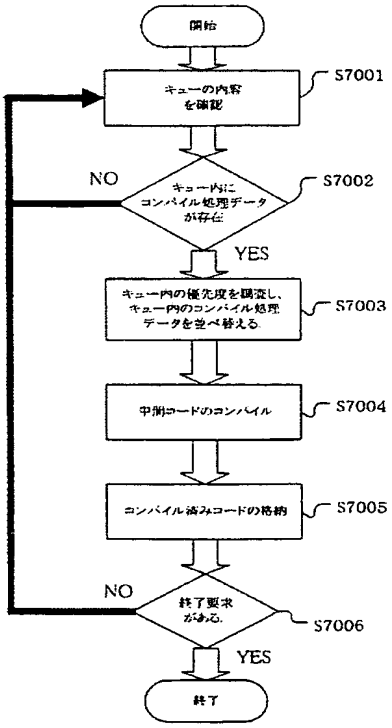
【図５】


コンパイラ
1008
コンパイル判定部
1006
5002
5000
5001
コンパイル処理の実行
コンパイル処理の投入

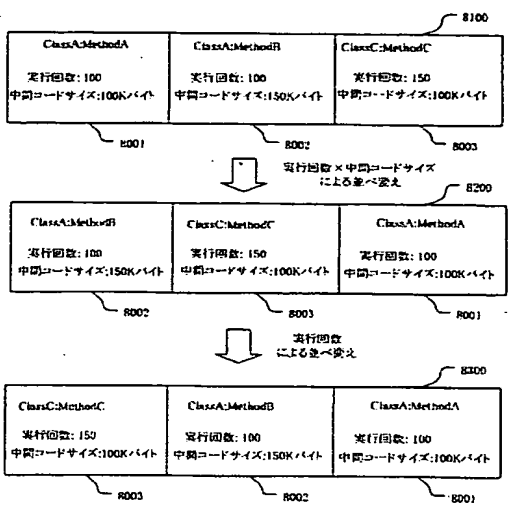
【図６】


6000
コンパイル対象を特定するための情報（クラス名、メソッド名等）
6001
プロファイル情報（呼出回数、実行回数、累積実行時間）
6002
静的解析情報（メソッドや中間コードのサイズ）
6003

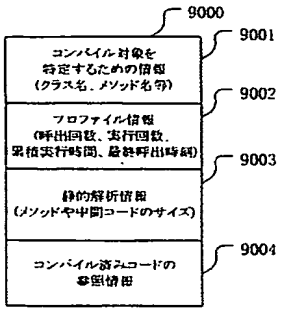
【図７】


開始
キューの内容を確認
S7001
キュー内にコンパイル処理データが存在
S7002
NO
YES
キュー内の優先度を調査し、キュー内のコンパイル処理データを並べ替える
S7003
中間コードのコンパイル
S7004
コンパイル済みコードの格納
S7005
終了要求がある
S7006
NO
YES
終了

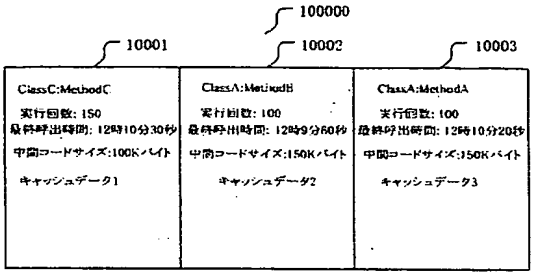
【図８】


8100
ClassA:MethodA
実行回数: 100
中間コードサイズ:100Kバイト
ClassA:MethodB
実行回数: 100
中間コードサイズ:150Kバイト
ClassC:MethodC
実行回数: 150
中間コードサイズ:100Kバイト
8001
8002
8003
実行回数×中間コードサイズ
による並べ変え
8200
ClassA:MethodB
実行回数: 100
中間コードサイズ:150Kバイト
ClassC:MethodC
実行回数: 150
中間コードサイズ:100Kバイト
ClassA:MethodA
実行回数: 100
中間コードサイズ:100Kバイト
8002
8003
8001
実行回数
による並べ変え
8300
ClassC:MethodC
実行回数: 150
中間コードサイズ:100Kバイト
ClassA:MethodB
実行回数: 100
中間コードサイズ:150Kバイト
ClassA:MethodA
実行回数: 100
中間コードサイズ:100Kバイト
8003
8002
8001

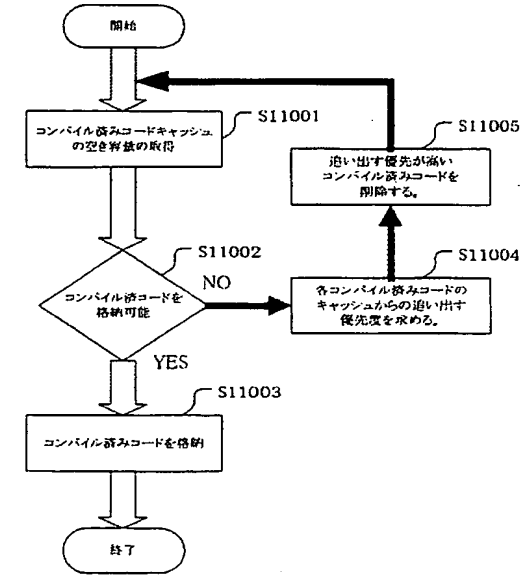
【図９】


9000
コンパイル対象を
特定するための情報
(クラス名、メソッド名等)
9001
プロファイル情報
(呼出回数、実行回数、
累積実行時間、最終呼出時刻)
9002
静的解析情報
(メソッドや中間コードのサイズ)
9003
コンパイル済みコードの
参照情報
9004

【図１０】


100000
10001
10002
10003
ClassC:MethodC
実行回数: 150
最終呼出時間: 12時10分30秒
中間コードサイズ:100Kバイト
キャッシュデータ1
ClassA:MethodB
実行回数: 100
最終呼出時間: 12時9分60秒
中間コードサイズ:150Kバイト
キャッシュデータ2
ClassA:MethodA
実行回数: 100
最終呼出時間: 12時10分20秒
中間コードサイズ:150Kバイト
キャッシュデータ3

【図１１】


開始
コンパイル済みコードキャッシュ
の空き容量の取得
S11001
S11005
追い出す優先が高い
コンパイル済みコードを
削除する。
S11002
NO
S11004
コンパイル済コードを
格納可能
各コンパイル済みコードの
キャッシュからの追い出す
優先度を求める。
YES
S11003
コンパイル済みコードを格納
終了

フロントページの続き

(72)発明者 高橋 克英
東京都千代田区丸の内二丁目２番３号 三菱電機株式会社内
(72)発明者 橘高 大造
東京都千代田区丸の内二丁目２番３号 三菱電機株式会社内
(72)発明者 清原 良三
東京都千代田区丸の内二丁目２番３号 三菱電機株式会社内
(72)発明者 中 邦博
東京都千代田区丸の内二丁目２番３号 三菱電機株式会社内
(72)発明者 渡辺 克也
東京都千代田区丸の内二丁目２番３号 三菱電機株式会社内

Ｆターム(参考) 5B081 AA09 CC11 CC21 DD01